UITS

UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES

Project Report

CSE 360 MPMC Lab

SUBMITED BY

NAME: MD. Rahmatulla Ashik

ID: 0432220005101007

BATCH: 52

SECTION: 6A1

DEPARTMENT: CSE

SUBMITED TO

Md. Ismail

Lecturer
B.Sc. (Engg) in CSE, Rajshahi University of Engineering & Technology (RUET)

Analysis Report: 8086 Assembly Language Calculator Program

1 Introduction

The 8086 assembly language program under analysis is a menu-driven calculator designed for the DOS operating system. It allows users to perform basic arithmetic operationsaddition, subtraction, multiplication, and divisionon two-digit numbers (0099) or exit the program. The program uses a small memory model and leverages DOS interrupts for input/output operations. This report aims to analyze the program's functionality, struc- ture, and limitations, providing insights into its implementation and potential improve- ments.

2 Methodology

The analysis was conducted by reviewing the provided 8086 assembly code, which in- cludes data and code segments, procedures, and interrupt-based I/O operations. The methodology involved:

- **Code Review**: Examining the structure, including memory model, data defini- tions, and procedures.
- **Functional Analysis**: Tracing the program flow from menu display to result output, focusing on input handling, arithmetic operations, and error management.
- **Limitation Identification**: Identifying constraints such as input range, error handling, and user interface limitations.
- **Testing Hypothetical Scenarios**: Evaluating the programs behavior with sample inputs (e.g., valid numbers, invalid inputs, division by zero).

The analysis assumes the program runs on an 8086-compatible DOS environment (e.g., DOSBox).

3 Implementation

The program is implemented in 8086 assembly language using the small memory model, with a 256-byte stack. Below is an overview of its components:

3.1 Data Segment

The data segment includes:

- **Messages**: Strings for the menu, input prompts, operation results, and error messages, formatted with carriage return (0dh) and line feed (0ah).
- **Variables**: num1 and num2 (8-bit) for input numbers, operator (8-bit) for the chosen operation, and result (16-bit) for storing computation results.

3.2 Code Segment

The code segment consists of the main procedure and three subroutines:

- **Main Procedure**: Initializes the data segment, runs a loop to display the menu, reads user input, performs calculations, and handles errors.
- **Display Menu**: Outputs the menu using DOS interrupt 21h, function 09h.
- **Read Number**: Reads two ASCII digits, converts them to a single 8-bit number (0099), and validates input (returns 0ffh for invalid input).
- **Display Number**: Converts the 16-bit result to ASCII digits, handles negative numbers, and prints them using a stack-based approach.

3.3 Program Flow

The program:

1. Displays a menu with options 15 (addition, subtraction, multiplication, division, exit).
2. Reads a single-character menu choice using int 21h, function 01h.
3. Prompts for two 2-digit numbers using read_number.
4. Performs the selected operation (add, sub, mul, div) and stores the result in result.
5. Displays the result using display_number or an error message for invalid inputs or division by zero.
6. Loops back to the menu unless the user selects exit (option 5).

3.4 Arithmetic Operations

- **Addition**: Adds num1 and num2 using add al, bl.
- **Subtraction**: Subtracts num2 from num1 using sub al, bl, with cbw for sign extension.
- **Multiplication**: Multiplies num1 and num2 using mul bl, storing the result in AX.
- **Division**: Divides num1 by num2 using div bl, checking for division by zero.

4 Input

The program accepts the following inputs:

- **Menu Choice**: A single character (15) to select an operation or exit.
- **Numbers**: Two 2-digit numbers (0099), entered as ASCII characters. For example, entering "45" is converted to the decimal value 45.

Input is read using DOS interrupt `21h`, function `01h` for single characters and a custom `read_number` procedure for numbers. The program validates digits (09) but does not handle extra characters or negative numbers.

4.1 Sample Input

```
Select the operation :
* Addition
* Subtraction
* Multiplication
* Division
* Exit
Enter your choice between (1 -5): 1
Enter first number (00 -99): 45
Enter second number (00 -99): 23
```

5 Output

The program outputs:

- The menu and prompts, formatted with newlines.
- The result of the arithmetic operation (e.g., "The Result of Addition is = 68").
- An error message ("Error: Invalid input or division by zero") for invalid inputs or division by zero.

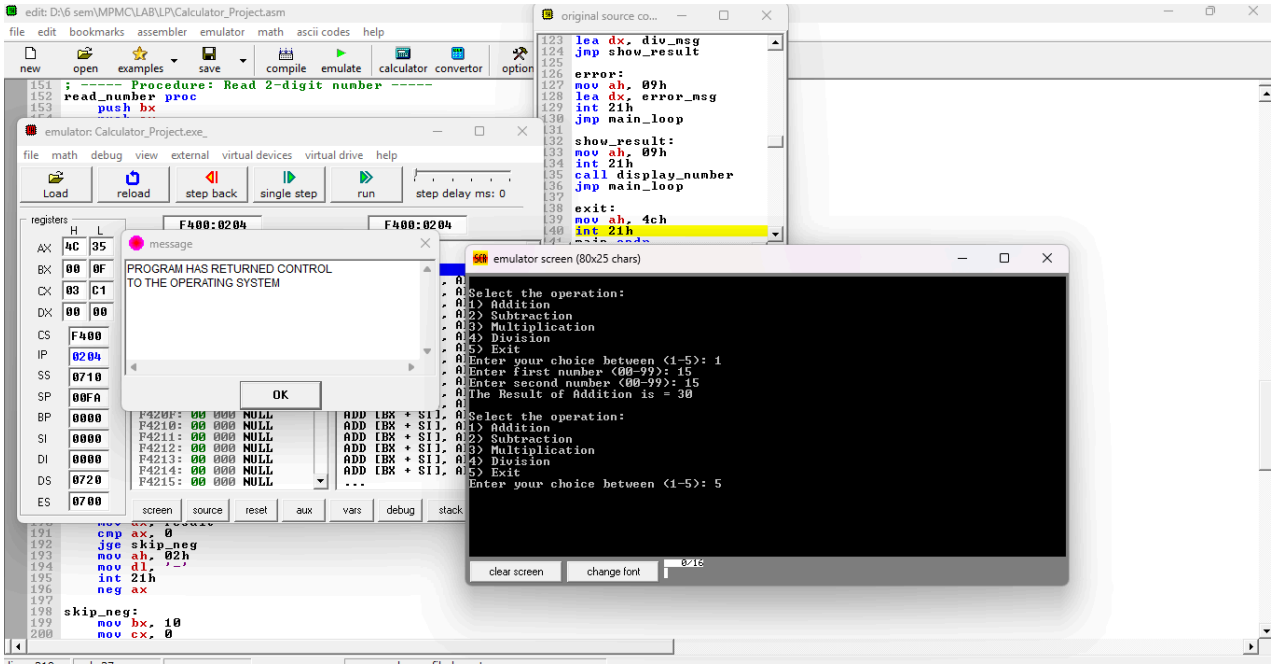
Results are displayed using `display_number`, which handles multi-digit and negative numbers. For example:

```
The Result of Addition is = 68
```

or, for subtraction:

```
The Result of Subtraction is = -10
```

5.1 Screenshot



6 Conclusion

The 8086 assembly calculator is a well-structured program that effectively performs basic arithmetic operations on two-digit numbers. Its modular design, with separate procedures for input, processing, and output, enhances maintainability. The program handles basic error cases (invalid inputs, division by zero) and provides a clear user interface. However, limitations include:

- Limited input validation (e.g., no handling of extra characters or negative inputs).
- Division outputs only the quotient, ignoring the remainder.
- Generic error messages that may confuse users.

6.1 Recommendations

- Enhance input validation to handle extra characters and negative numbers.
- Display division remainders for completeness.
- Use specific error messages for different error types.
- Clear the keyboard buffer to prevent input interference.
- Add a pause after displaying results to improve user experience.

This program serves as a solid foundation for a basic calculator and could be extended with additional features, such as support for larger numbers or more operations, with minor modifications.